

MULTICORE DSP

FROM ALGORITHMS TO REAL-TIME
IMPLEMENTATION ON THE TMS320C66x SoC

NAIM DAHNOUN



WILEY

This text offers special coverage of the fundamentals of multicore DSP for implementation on the TMS320C66x SoC

This content provides readers with an understanding of the TMS320C66x SoC as well as its constraints. It offers critical analysis of each element, which not only broadens their knowledge of the subject, but aids the reader in gaining a better understanding of how these elements work so well together.

Written by Texas Instruments' First DSP Educator Award winner, Naim Dahnoun, the text teaches readers how to use the development tools, take advantage of the maximum performance and functionality of this processor and have an understanding of the rich content which spans from architecture, development tools and programming models, such as OpenCL and OpenMP, to debugging tools. The text also covers various multicore audio and image applications in detail and is supplemented with:

- A rich set of tested laboratory exercises and solutions
- Audio and Image processing applications source code for the Code Composer Studio (integrated development environment from Texas Instruments)
- Multiple tables and illustrations

With its rich content of twenty chapters, *Multicore DSP: From Algorithms to Real-time Implementation on the TMS320C66x SoC* is a rare and much-needed source of information for undergraduates and postgraduates in the field that allows them to make real-time applications work in a relatively short period of time. This content is also incredibly beneficial to hardware and software engineers involved in programming real-time embedded systems.


Naim Dahnoun is Reader in Teaching and Learning in Signal Processing in the Faculty of Engineering at the University of Bristol, UK.



Cover Design: Wiley
Cover Image: © matejmo/Gettyimages

www.wiley.com/go/dahnoun/multicoresdp

WILEY

 Also available
as an e-book



DAHNOUN

MULTI-CORE DSP



WILEY

Multicore DSP

From Algorithms to Real-time Implementation
on the TMS320C66x SoC

Naim Dahnoun
University of Bristol
UK

WILEY

This edition first published 2018
© 2018 John Wiley & Sons Ltd

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by law. Advice on how to obtain permission to reuse material from this title is available at <http://www.wiley.com/go/permissions>.

The right of Naim Dahnoun to be identified as the author of this work has been asserted in accordance with law.

Registered Office(s)

John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

Editorial Office

The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

Limit of Liability/Disclaimer of Warranty

While the publisher and authors have used their best efforts in preparing this work, they make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives, written sales materials or promotional statements for this work. The fact that an organization, website, or product is referred to in this work as a citation and/or potential source of further information does not mean that the publisher and authors endorse the information or services the organization, website, or product may provide or recommendations it may make. This work is sold with the understanding that the publisher is not engaged in rendering professional services. The advice and strategies contained herein may not be suitable for your situation. You should consult with a specialist where appropriate. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor authors shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

Library of Congress Cataloging-in-Publication data applied for

ISBN: 9781119003823

Cover design by Wiley

Cover image: © matejmo/Gettyimages

Set in 10/12pt Warnock by SPi Global, Pondicherry, India

Printed in Singapore by C.O.S. Printers Pte Ltd

10 9 8 7 6 5 4 3 2 1

Multicore DSP

*I dedicate this book to my children
Zahra, Yasmin and Riyad
and in memory of my parents*

Preface

Today's many applications, such as medical, high-end imaging, high-performance computing and core networking, are facing increasing challenges in terms of data traffic, processing power and device-to-device communication. These put a high demand on the processor(s) and associated software and lead to processor manufacturers sustaining Moore's law by introducing multicore processors. Texas Instruments, with its leading-edge technology, introduced the multicore System-on-Chip (SoC) architecture family of processors to address these issues. As will be shown in this book, Texas Instruments introduced innovations at many levels, such as: powerful CPUs that support both fixed- and floating-point arithmetic (instruction by instruction) that can achieve more than 40G multiplications/core, a Navigator that enables direct communication between cores and memory access that removes data movement bottlenecks, a Hyperlink interface and advanced development tools.

The challenge is not only how many cores you can put on a piece of silicon, the processing power of each core and how fast they can communicate, but also in the programming model and ease of use. Unfortunately, programming models are not developed sufficiently to handle several cores. The improvement in performance gained by the use of a multicore processor depends very much on the application and software used. C and C++, which are commonly used in embedded systems, do not support partitioning and, therefore, porting sequential code to multicore is not trivial. In this book, it will be shown this complexity is alleviated by using: OpenMP, which is an Application Programming Interface (API) that supports multiplatform shared multiprocessing programming in C, C++ and Fortran; Open Computing Language (OpenCL); or the Inter-Processor Communication (IPC).

This book will help to innovate by making the reader understand the KeyStone SoC architectures, the development tools including debugging and various programming models with tested examples, and also help to broaden the knowledge by critically analysing each element (see Table of Contents) and understanding how these elements are working together. With the sheer number of practical examples and references provided, the reader will be able to quickly develop applications, take advantage of maximum performance and functionality of the processors, be able to easily use the tools to develop and debug applications and find the relevant references to pertinent material. Real-time multicore audio and video applications are provided. Applications will be based on TI's Multicore Software Development Kit (MCSDK), hand-optimised code, OpenMP, OpenCL and IPC.

Due to the sheer amount of documentation available, some information is either referred to or reproduced to avoid discontinuity and misinterpretation.

This book is divided into 20 chapters. Chapters 1 to 15 deal with the hardware and software issues, and Chapters 16 to 20 deal with applications. Most of the concepts are backed up with laboratory experiments and demos that have been thoroughly tested.

Chapter 1 Introduction: This introductory chapter provides the reader with general knowledge on multicore processors and their applications; gives a brief comparison between digital signal processor (DSP) SoCs, field-programmable gate arrays (FPGAs), graphic processors and CPUs; illustrates the challenges associated with multicore; and provides an up-to-date TMS320 road-map showing the evolution of TI's DSP chips in terms of processing power.

Chapter 2 The TMS320C66x architecture overview: This chapter comprehensively describes the TMS320C66x architecture. This includes a detailed description of the DSP CorePacs and an overview of the peripherals, and it introduces some useful instructions and an overview of the memory organisation.

Chapter 3 Software development tools and the TMS320C6678 EVM: This chapter describes the software development tools that are required for testing the applications used in this book. It provides a step-by-step guide to the installation and use of the Code Composer Studio (CCS).

Chapter 4 Numerical issues: This chapter explains how fixed and floating points are represented and how to handle binary arithmetic. It provides examples showing how to display various data formats using the CCS.

Chapter 5 Software optimisation: This chapter discusses the different levels of optimisation for multicore and shows how code can be optimised for a DSP core. This chapter also shows how to use intrinsics and interface C language with intrinsics and assembly code. Multiple examples showing how to optimise code by hand and using the tools are provided.

Chapter 6 The TMS320C66x interrupts: This chapter shows how the interrupt controller events and the Chip-level Interrupt Controller work and how to program them to respond to events. The examples given use the general-purpose input-output (GPIO) pins to provide the interrupts.

Chapter 7 Real-time operating system: TI-RTOS: This chapter is divided into three main sections: (1) a real-time scheduler that is composed of the hardware and software interrupts, the task, the idle, clock and timer functions, synchronisation and events; (2) dynamic memory management; and (3) laboratory experiments.

Chapter 8 Enhanced Direct Memory Access (EDMA3) Controller: This chapter describes in detail the operation of the EDMA and provides examples with simple transfer, chaining transfer and linked transfer.

Chapter 9 Inter-Processor Communication (IPC): This chapter explains the need for IPC and describes the notify module, the messageQ, the ListMP module, the Multi-processor Memory Allocation, the transport mechanism and laboratory examples.

Chapter 10 Single and multicore debugging: This chapter introduces the need for debugging and describes the debug architecture that includes trace, Advanced Event Triggering and the Unified Breakpoint Manager. This chapter also describes the Unified Instrumentation Architecture, debugging with the System Analyzer tools, instrumentation with TI-RTOS and CCS and laboratory experiments.

Chapter 11 Bootloader for Keystone I and Keystone II: This chapter introduces the boot process for both the KeyStone I and KeyStone II, and provides laboratory experiments for both devices.

Chapter 12 Introduction to OpenMP: This chapter introduces the concept behind OpenMP and divides the content into three main sections: (1) work sharing, (2) data sharing and (3)

synchronisation. Various examples with both KeyStone I and II are provided. For the KeyStone II, an example is implemented with OpenMP with the accelerator model.

Chapter 13 Introduction to OpenCL for the KeyStone II: In this chapter, another programming model called Open Computing Language (OpenCL) is introduced. This chapter will emphasise the OpenCL for the KeyStone rather than other devices. This chapter will show that OpenCL is easy to use since the programmer does not need to deal with details of communication between DSP cores or between the ARM and the DSP, which may be a daunting task.

Chapter 14 Multicore Navigator: This chapter shows how the Multicore Navigator can provide a high-speed packed data transfer to enhance CorePac to accelerator/peripheral data movements, core-to-core data movements, inter-core communication and synchronisation without loading the CorePacs. Examples are also provided.

Chapter 15 FIR filter implementation: The purpose of this chapter is twofold. Primarily, it shows how to design an FIR filter and implement it on the TMS320C66x processor; and, secondly, it shows how to optimise the code as discussed in Chapter 3. This chapter discusses the interface between C and assembly, how to use intrinsics, and how to put into practice material that has been covered in the previous chapters.

Chapter 16 IIR filter implementation: This chapter introduces the IIR filters and describes two popular design methods: the bilinear and the impulse invariant methods. Step by step, this chapter shows the procedures necessary to implement typical IIR filters specified by their transfer functions. Finally, this chapter provides complete implementation of an IIR filter in C language, assembly and linear assembly, and shows how to interface C with linear assembly.

Chapter 17 Adaptive filter implementation: This chapter starts by introducing the need for an adaptive filter in communications. It then shows how to calculate the filter coefficients using the mean squared error (MSE) criterion, exposes the least mean squares (LMS) algorithm and, finally, shows how the LMS algorithm is implemented in both C and assembly.

Chapter 18 FFT implementation: This chapter shows a derivation of an FFT algorithm and shows its implementation in C language. To improve the performance, the ping-pong EDMA has been used.

Chapter 19 Hough transform: This chapter shows the basic mathematics behind the Hough transform for detecting straight lines and how to implement it. This chapter also shows how to increase the performance by looking at the algorithm and minimising the number of operations required, and how to use the graphical display using the Code Composer Studio.

Chapter 20 Stereo vision implementation: This chapter shows the principle behind the stereo vision system and highlights different levels of optimisations for achieving real-time performance. Some techniques for reducing the processing time for calculating the disparity values for automotive applications are also introduced.

Acknowledgements

I didn't expect that writing another book would be challenging, considering that I have written previous material. This was mainly due to the fast-moving technology and because systems are getting more complex, but I had to keep up with it. Putting all knowledge gained in a simple form and sharing it give me great satisfaction.

My first thanks go to the Engineer to Engineer (e2e.ti.com) community that I found extremely generous and very helpful.

I am indebted to Cathy Wicks, Jocken Schyma, Jason Brand, Garry Clarkson and Rogerio Almeida, the key motivators and initiators for writing this book.

I would like to express my gratitude to the following reviewers for their insightful suggestions and comments: Pekka Varis, Dave Bell, Eric Stotzer, John Smrstik, Jennifer Stadelmann, Ran Katzur, Steve Preissig, Naga Chandrashekar, Greg Peake and Filip Moerman.

I owe my thanks to Professor John Rarity, Professor Andy Nix, Professor Dave Cliff, Professor David May, Dr Richard Nock, Dr Ross Xi, Dr Sergey Vityazev, Mr James Webley and all my colleagues at the Faculty of Engineering, University of Bristol, for their encouragement and support. I also thank all my students, and in particular Scott Tancock, Han Cui, Aleksandar Stanoev, Victor Prokhorov, Aliaksei Mikhailiuk, Akmal Ahmed, Ranger Fan, Elliott Worsey and Charles Khoury who were always eager to explore challenging issues, verify and test applications.

My thanks to my friends Professor Hamdani Abdelsalam, Judge Shamin and his wife Shabina, Hernandez Paul and his wife Khansa, Mr Baris Tanyeri, Mr Karaborek Yunus, Dr Tila Fai, Mr Ghoul Amar and Dr Seti Chenafa for their continuous support and friendship.

To my friend Gene Frantz, 'the father of DSP', who wrote the Foreword to my book and who has been a great inspiration to me.

Finally, many thanks to Ruth Thomas for her kindness in reading the whole manuscript and providing feedback, and Alex King and Preethi Belkese from Wiley who were very easy to deal with, encouraging and supportive.

Naim Dahnoun
Naim.Dahnoun@Bristol.ac.uk

Foreword

Having spent my professional career introducing the digital signal processing (DSP) technology and associated products to the industry, and now as a Professor in the practice at Rice University, I continue looking for the next use or user of DSP. One of the high points of my career has been working with professors and authors who are preparing the next generation of talented engineers.

I have known Naim for about 20 years, before he wrote his first and popular book entitled 'Digital Signal Processing Implementation: Using the TMS320C6000TM Platform', which I reviewed. Since then, DSP processors have evolved into advanced heterogeneous multicore processors that are hard to program. To extract maximum performance, programmers need to master not only the applications that must be implemented but also the processor's hardware and supporting software. For instance, many programming models such as Message Passing Interface (MPI), Open Multi-Processing (OpenMP), Open Computing Language (OpenCL) and Inter-Processor Communication (IPC) have been introduced to ease development, in addition to the operating systems. Each of these programming languages is implemented differently by different device manufacturers, and each of these programming languages is covered in separate books. To make the best use of these programming models, one needs to compare and contrast them for a specific application. This book covers most of these programming models and gives the reader a good starting point.

This book is rich in its well-structured content and is worthy of deep and reflective reading. It starts by highlighting solutions of some problems on multicore processors, and then focusses on multicore DSPs. To gain maximum performance, this book provides details at the assembly and the linear assembly levels, and then shows how this could be achieved by using the appropriate compiler switches to save development time, increase portability and reduce maintenance. The book then tackles IPC, OpenMP, OpenCL and the Navigator to ease programming the Multicore DSP, and it provides a rich set of practical examples for both the KeyStone I and the KeyStone II platforms.

Debugging is as important as programming itself, especially in large and complex applications. With this in mind, silicon manufacturers have heavily invested in both hardware and software debugging, and in this book, Naim recognises the need to simplify its use.

In addition to hardware and development software, this book also shows how to implement main signal-processing algorithms such as FIR, IIR, adaptive filters, Hough transform, FFTs and disparity calculation for stereo vision applications.

There is no doubt that this book, with its comprehensive content, will provide the reader with knowledge and inspiration that will allow him or her to experiment and maybe push the boundaries even further.

Gene Frantz

About the Companion Website

Don't forget to visit the companion website for this book:

www.wiley.com/go/dahnoun/multicoresdp



There you will find valuable material designed to enhance your learning, including:

- 1) Appendix 1: Creating a Virtual Machine
- 2) Appendix 2: Software Directory
- 3) Appendix 3: Software updates
- 4) Exercises and Solutions
- 5) Source codes

Scan this QR code to visit the companion website:



Contents

Preface	<i>xviii</i>
Acknowledgements	<i>xxi</i>
Foreword	<i>xxii</i>
About the Companion Website	<i>xxiii</i>

1	Introduction to DSP	1
1.1	Introduction	1
1.2	Multicore processors	3
1.2.1	Can any algorithm benefit from a multicore processor?	3
1.2.2	How many cores do I need for my application?	5
1.3	Key applications of high-performance multicore devices	6
1.4	FPGAs, Multicore DSPs, GPUs and Multicore CPUs	8
1.5	Challenges faced for programming a multicore processor	9
1.6	Texas Instruments DSP roadmap	10
1.7	Conclusion	11
	References	12
2	The TMS320C66x architecture overview	14
2.1	Overview	14
2.2	The CPU	15
2.2.1	Cross paths	16
2.2.1.1	Data cross paths	17
2.2.1.2	Address cross paths	18
2.2.2	Register file A and file B	20
2.2.2.1	Operands	20
2.2.3	Functional units	21
2.2.3.1	Condition registers	21
2.2.3.2	.L units	22
2.2.3.3	.M units	22
2.2.3.4	.S units	23
2.2.3.5	.D units	23
2.3	Single instruction, multiple data (SIMD) instructions	24
2.3.1	Control registers	24
2.4	The KeyStone memory	24
2.4.1	Using the internal memory	27
2.4.2	Memory protection and extension	29
2.4.3	Memory throughput	29